48024 Applications Programming

Assignment 2

Topics: OO design, GUI, MVC, tables, lists
Objectives: This assignment supports objectives 3 - 5
Due date: 5pm Friday 10th of June 2016
Weight: 20%

1. Individual work

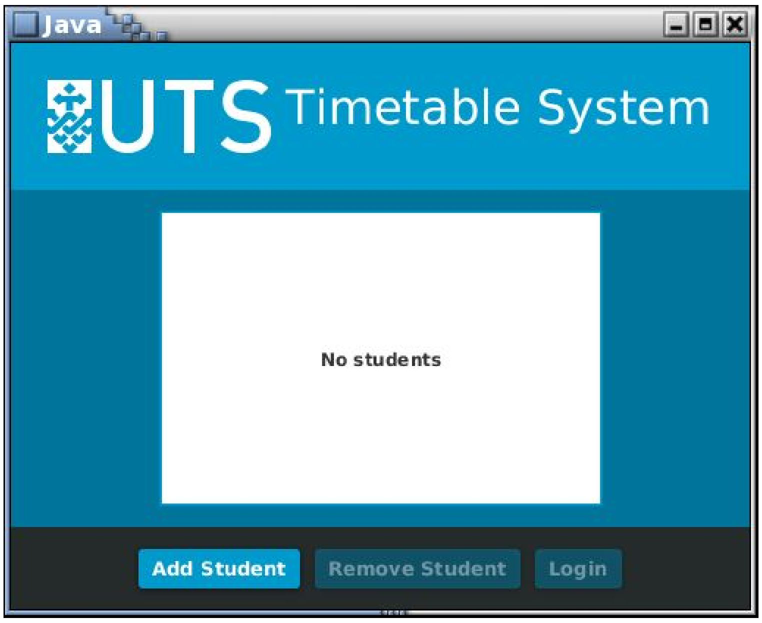
All work is individual. You may discuss ideas, approaches and problems, but you should write every line of code yourself except for code copied from the lecture notes, lecture code or lab code. More information about Academic Misconduct can be found at: http://www.gsu.uts.edu.au/rules/student/section-16.html

2. Specification

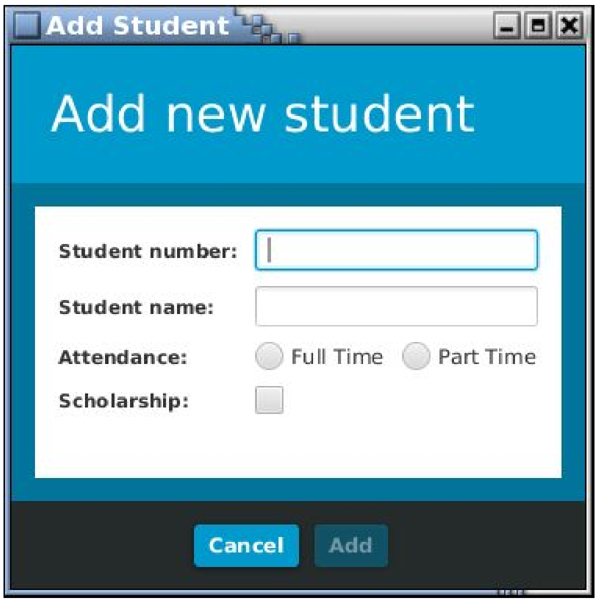
After a successful trial run of the Assignment 1 timetable system, UTS has decided it would like you to build a graphical user interface for the application using JavaFX.

Functionality

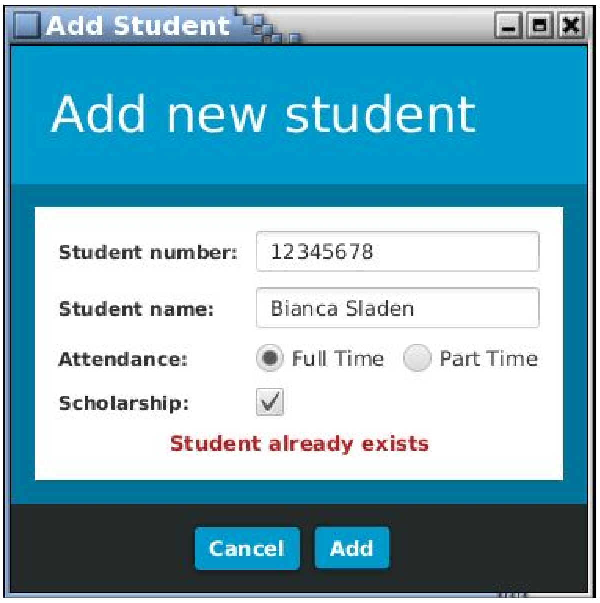
When the application is launched, the "University" window appears which shows a list of students, or the placeholder "No students" if there are no students (as is the case initially):



Clicking “Add Student” brings up the “Add Student” window with a data entry form:



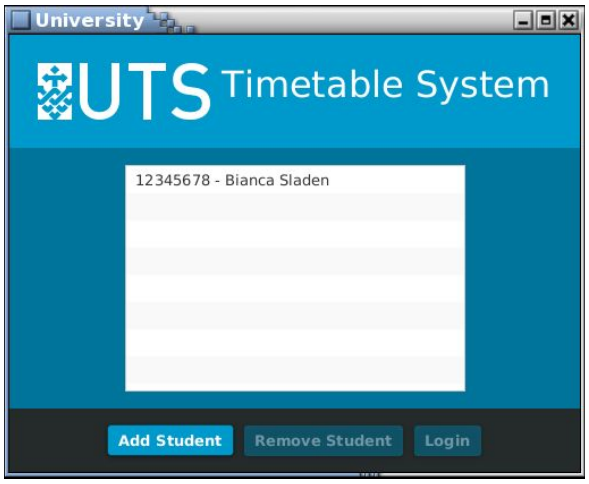
Student number:
Student name:
Attendance: Full Time Part Time
Scholarship:



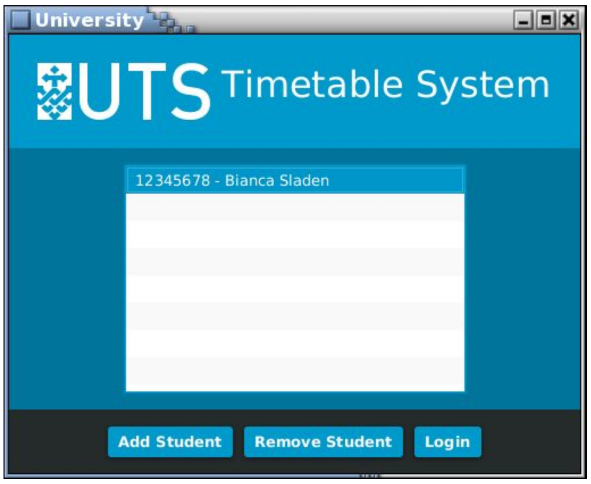
Student number: 12345678
Student name: Bianca Sladen
Attendance: [✓ Full Time] Part Time
Scholarship: ✓
Student already exists

Clicking “Cancel” closes the window without adding a new student. The “Add” button is disabled while the student number is blank, or the student name is blank, or an attendance mode has not been chosen. Clicking “Add” will result in the error “Student already exists” if a student with the given number already exists in the university.

Otherwise, clicking “Add” will add a new student to the university using the data entered, and the “Add Student” window is closed. The “University” window instantly displays the new student in the list:



12345678 - Bianca Sladen

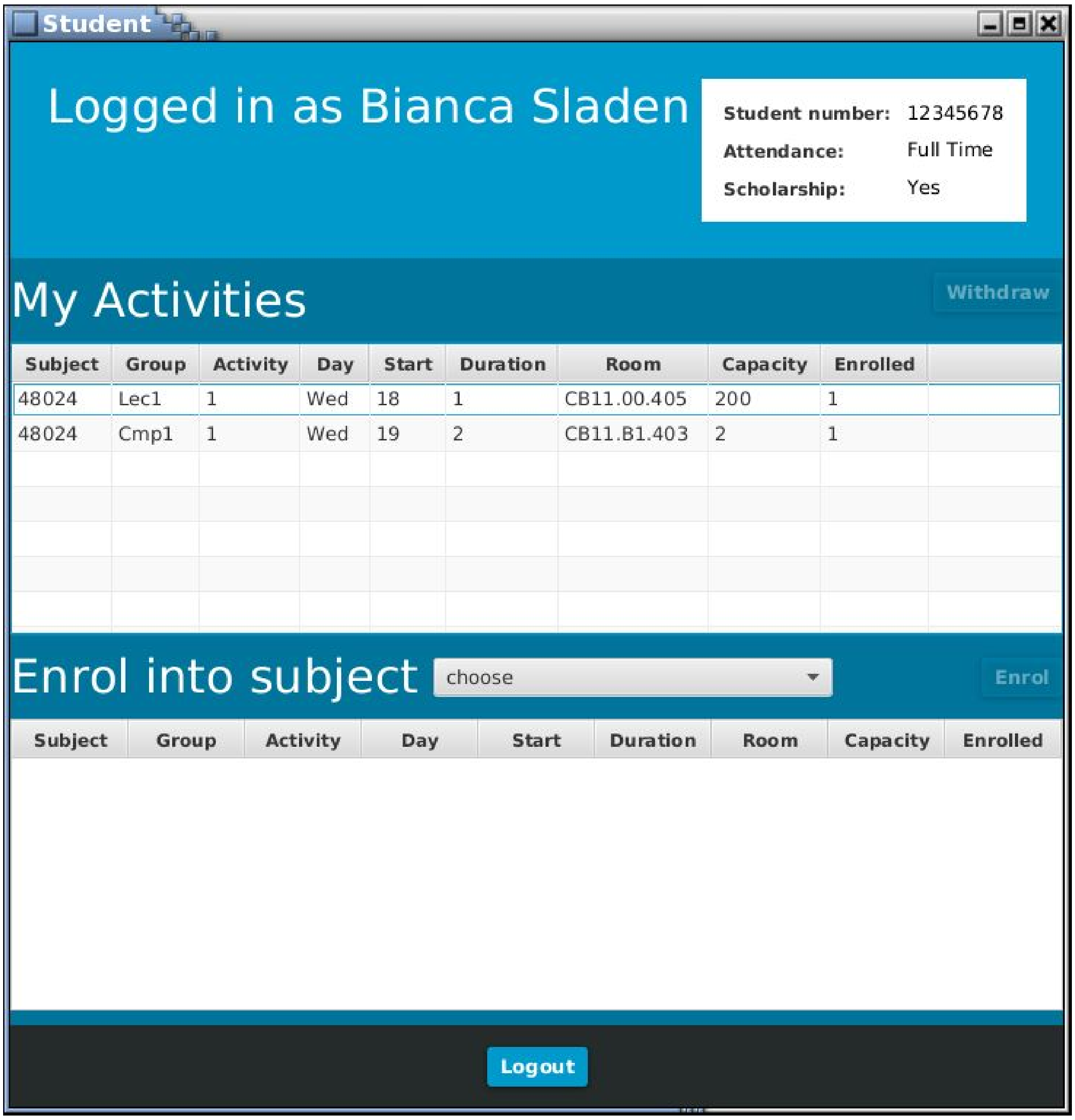


12345678 - Bianca Sladen

The “Remove Student” and “Login” buttons are enabled only when a student has been selected in the list.

Clicking “Remove Student” will remove the student from the university and withdraw the student from all activities in which that student is enrolled. The user interface is immediately updated to reflect this change.

Selecting a student and clicking “Login” will open the “Student” window:



Logged in as Bianca Sladen

Student number:	12345678
Attendance:	Full Time
Scholarship:	Yes

My Activities

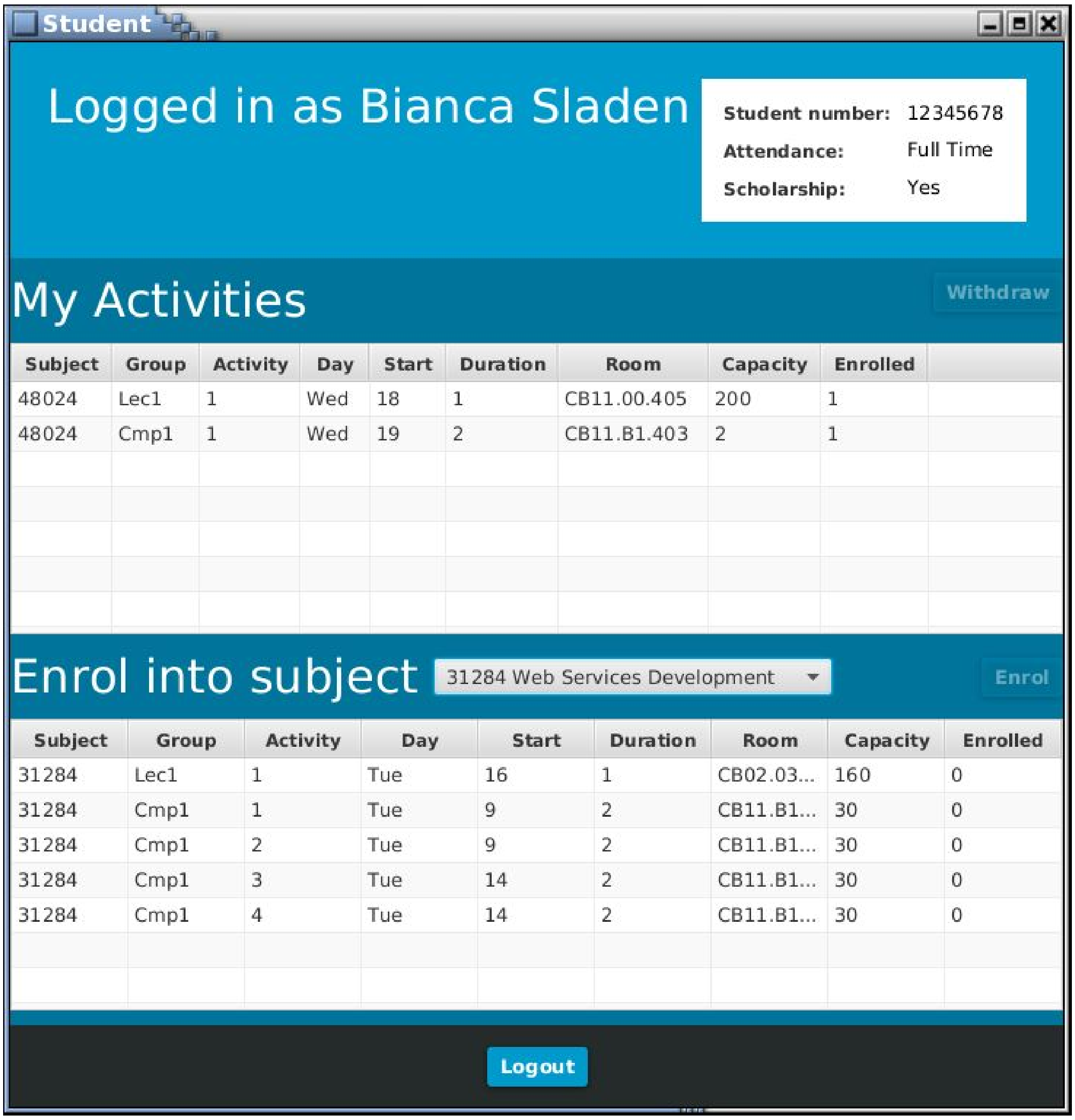
Subject	Group	Activity	Day	Start	Duration	Room	Capacity	Enrolled
48024	Lec1	1	Wed	18	1	CB11.00.405	200	1
48024	Cmp1	1	Wed	19	2	CB11.B1.403	2	1

Enrol into subject

Subject	Group	Activity	Day	Start	Duration	Room	Capacity	Enrolled

The student’s name, number, attendance and scholarship is shown at the top. The attendance is shown as either “Full Time” or “Part Time”. The scholarship is shown as either “Yes” or “No”. The “My Activities” table displays all the activities that the student is currently enrolled into, or the placeholder “Not enrolled in any activities” if the student is not enrolled in any activities.

To enrol into an activity, the user first chooses a subject from the ComboBox. After selecting a subject, the table below the ComboBox presents a list of activities in that subject:



Logged in as Bianca Sladen

Student number:	12345678
Attendance:	Full Time
Scholarship:	Yes

My Activities

Subject	Group	Activity	Day	Start	Duration	Room	Capacity	Enrolled
48024	Lec1	1	Wed	18	1	CB11.00.405	200	1
48024	Cmp1	1	Wed	19	2	CB11.B1.403	2	1

Enrol into subject

31284 Web Services Development

Subject	Group	Activity	Day	Start	Duration	Room	Capacity	Enrolled
31284	Lec1	1	Tue	16	1	CB02.03...	160	0
31284	Cmp1	1	Tue	9	2	CB11.B1...	30	0
31284	Cmp1	2	Tue	9	2	CB11.B1...	30	0
31284	Cmp1	3	Tue	14	2	CB11.B1...	30	0
31284	Cmp1	4	Tue	14	2	CB11.B1...	30	0

The user may then select an activity from that table and click “Enrol”. The “Enrol” button is enabled only when an activity has been selected and there is an available seat in that activity and the student isn’t already enrolled in that activity. Clicking the “Enrol” button results in the student being enrolled into that activity, and being withdrawn from any activity in that subject with the same group as the new activity being enrolled into. The user interface is immediately updated to reflect the new system state. In particular, the “Enrolled” column for the selected activity increases by 1, and the activity appears in the “My Activities” table. If the student was first withdrawn from another activity, the “Enrolled” column for that activity decreases by 1. Buttons should also be appropriately enabled or disabled based on the current state (see above for the when to disable the “Enrol” button and below for when to disable the “Withdraw” button).

To withdraw from an activity, the user selects an activity from the “My Activities” table and clicks “Withdraw”. The “Withdraw” button is enabled only when an activity has been selected from the “My Activities” table. Upon clicking “Withdraw”, the user interface is immediately updated to reflect the new system state.

Clicking “Logout” closes the “Student” window.

Layout

Every window is divided vertically into 3 overall parts: a header section, a content section, and a footer section.

- The “University” window shows:
    - A header containing the UTS logo (downloadable from http://www.uts.edu.au/sites/default/files/logo_2.png) and the heading text “Timetable System”.
    - A content section containing a ListView with a preferred width of 300 pixels and a preferred height of 200 pixels. The ListView is centred horizontally within the content section.
    - A footer containing 3 buttons centred horizontally.
- The “Add Student” window shows:
    - A header containing the heading text “Add new student”.
    - A content section containing a form laid out in a GridPane. The grid has 5 rows containing 4 Labels, 2 TextFields, 2 RadioButtons, 1 CheckBox and a Text to display the error message. The error message spans 2 columns and is centred horizontally.
    - A footer containing 2 buttons centred horizontally.
- The “Student” window shows:
    - A header containing the heading text “Logged in as <student name>” at the top left, and a GridPane at the top right. The GridPane contains 3 Labels in the left column and 3 Texts in the right column.
    - A content section containing two sub-sections:
        - The first section begins with the heading text “My Activities” aligned to the left, and a button aligned to the right. Below both is a TableView.
        - The second section begins with the heading text “Enrol into subject” followed by a ComboBox, both aligned to the left, and a button aligned to the right. Below these is a TableView.
    - A footer containing a button centred horizontally.

Style

The following CSS colours are used:

- `#0099cc` is used as the background colour of the header sections and the buttons.
- `#00749b` is used as the background colour of the content sections.
- `#252b2b` is used as the background colour of the footer sections.
- `#333333` is used as the colour of labels.
- `#ffffff` is used as the colour of heading text and button text.
- `FIREBRICK` is used as the colour of error messages.
- `#00bfff` is used as the background colour of a button during the hover state.

The following fonts are used:

- Heading text is Arial with font size 32px.
- Labels use font size 12px.
- Label text, Button text and error text is bold.
- Default fonts are used elsewhere.

Spacing and padding is also important. While exact measurements are not given in this specification, your application should include spacing and padding roughly equivalent to the screenshots above.

Code

Your solution must satisfy the following code requirements:

- Your solution must follow an MVC architecture.
- Your package structure should place the models in a package named “model”, the views in a package named “view” and the controllers in a package named “controller”.
- The models must notify the views of changes by correctly applying the JavaFX property patterns and observable lists. Any model data that can change must be observable. Any model data that never changes does not need to be observable.
- The attendance mode must be stored in the student’s attendance String property. The value “ft” should be stored to represent Full Time, and the value “pt” should be stored to represent Part Time.
- Each view must be defined in FXML.
- The “Student already exists” error message must be implemented as an exception. The model should throw an exception, and the controller must catch the exception and show the error message on the view.

3. Skeleton Code

Download the skeleton code for Assignment 2 from the Assignment 2 assessment page on PLATE (https://plate.it.uts.edu.au/). Your solution must be based on this skeleton code.

Included in the skeleton code is a text file called `progress.txt` which must be filled in by you as you progress through the assignment (see “PLATE marking” and “Submission and Return” below).

4. Expected workload

The time to do the assignment to a distinction level (i.e. a mark between 75% to 84%) has been estimated at 15 hours for a student of average ability who has completed all the tutorial and lab exercises.

5. Online support

The Assignment 2 discussion board has been set up so that students can ask questions, and other students can reply. I will post a reply only if I think the student response was wrong, or in the case of correcting a mistake in the assignment specification.

You must not post Java code to the discussion board. The board is there to help you, not to provide the solution. Posting your code is academic misconduct and will reported. Each time this rule is violated, I will delete the code and post a comment of the form: “Strike 1: Posting code”. After 3 strikes, the discussion board will be deleted because it did not work.

FAQs (Frequently Asked Questions) and their answers are posted on UTSOnline in Assignments/2/faq. If you have a question, check the FAQ first; it may already be answered there. You should read the FAQ at least once before you hand in your solution, but to be safe check it every couple of days. Anything posted on the FAQ is considered to be part of the assignment specification. The FAQ will be frozen (no new entries) two days before the due date; no questions will be answered after it is frozen.

If anything about the specification is unclear or inconsistent, contact me and I will try to make it clearer by replying to you directly and posting the common questions and answers to the FAQ. This is similar to working on the job, where you ask your client if you are unsure what has to be done, but then you write all the code to do the task. Email Ryan.Heise@uts.edu.au to ask for any clarifications or corrections to the assignment.

6. PLATE marking

READ THIS ENTIRE SECTION CAREFULLY

PLATE cannot test the behaviour and appearance of a graphical user interface on its own. Instead, the mark reported by PLATE is based on a progress report that is filled in and submitted by YOU and double checked by staff.

Included in the skeleton code is a file called `progress.txt` which you must fill out as you progress through the assignment. A sample of the beginning of this file is shown below:

Launch the application
[?] The UTS Logo is shown
[?] The "Timetable System" heading is shown

[?] "No students" is shown in the empty list
[?] "Add Student" appears and is enabled
[?] "Remove Student" appears and is disabled
[?] "Login" appears and is disabled
[?] The layout arranges the header, content and footer sections vertically
[?] All colours and fonts are correct
[?] The spacing and padding look like the screenshots
Click "Add Student"
[?] The "Add Student" window is shown
[?] The "Add new student" heading is shown
[?] The student number label and text field are shown
[?] The student name label and text field are shown
[?] The attendance label, "Full Time" radio button and "Part Time" radio button are shown
[?] The scholarship label and checkbox are shown
[?] The "Cancel" button is shown and enabled
[?] The "Add" button is shown and disabled
[?] The layout arranges the header, content and footer sections vertically
[?] The form is laid out in a GridPane
[?] All colours and fonts are correct including the error message
[?] The error message is included in the GridPane, centred horizontally spanning 2 columns
[?] The spacing and padding look like the screenshots
Click "Cancel"
[?] The "Add Student" window is closed

This file tells you how to test the functionality of your application. Here is how it works:

- Each line without a [?] is an action you need to perform on your application.
- Each line with a [?] is verification point that you must check. You must replace the [?] by either a [y] (for yes) or a [n] (for no) to record whether or not your application behaves correctly in that respect.

For example, you should first launch the application, and then you should then check that the UTS Logo is shown, the "Timetable System" heading is shown, and so on. If your application does not display the logo, but does display the heading "Timetable System", then you should fill in that part of the `progress.txt` file as follows:

Launch the application
[n] The UTS Logo is shown
[y] The "Timetable System" heading is shown

If a verification point lists multiple things to check, they must all be true for you to give a [y] answer. For example:

[?] "Add Student" appears and is enabled

This verification point can only be answered [y] if BOTH the "Add Student" button appears AND it is enabled. Before your final submission, you should re-check every answer you gave, to ensure that your answers are accurate.

There are 85 verification points like this to fill in. Each [y] answer contributes a potential 1 mark to your total, which makes up a maximum of 85 marks out of 100. The remaining 15 marks are decided when a staff member looks at your code after the due date. The mark reported by PLATE when you submit is not final, and penalties may be applied to your code for reasons that include fooling or spoofing PLATE (See "Submission and return" and "Marking Scheme" below).

So that we have a record of your progress, you are required to submit your project (including the updated `progress.txt` file) to PLATE regularly. Serious penalties apply if you do not submit your progress in small increments (See "Marking Scheme" below).

If you are not sure how to implement a certain feature, in many cases it is possible to skip that feature and come back to it later if it is non-essential to testing the following features. For example, style and layout can be refined after you get the application working. Correct enabling and disabling of buttons can be addressed later. Radio buttons and checkboxes can be done later. In such cases where it is possible to continue testing your app after skipping a particular verification point, you are permitted to just enter [n] for that verification point and continue on with the next one.

However, you must perform all actions mentioned in progress.txt in sequence before attempting to check any particular verification point. For example, you cannot just "Launch the application" and then skip to the last verification point shown above and answer [y] that the "Add Student" window is closed. In order to be eligible to answer [y] for that verification point, you must have first also performed the action to open the "Add Student" window, and then also performed the action to click the "Cancel" button. Only then, if the "Add Student" window is closed, are you eligible to answer [y] that this window was closed.

7. Submission and return

Your solution is to be submitted to PLATE at https://plate.it.uts.edu.au/ under Applications Programming / Assessments / Assignment 2. Your assignment should be submitted as a JAR file that includes:

- All Java source files required to compile your assignment.
- All FXML, CSS and image files required to run your assignment.
- The progress.txt file at the top level of your project directory structure.

A provisional mark of up to 85 marks is generated immediately from your `progress.txt` file each time you submit to PLATE. However, marking the code (worth 15 marks), analysis of spoofing, cheating and plagiarism is done in the two weeks following the due date. If you are suspected of Academic Misconduct, I will forward your case to the Misconduct Committee and will notify you by email.

There is no scheduled late submission period. An extension of up to one week may be given by the subject coordinator before the due date; you have to supply documentary evidence of your claim. An extension CANNOT be given after the due date.

You may also apply for special consideration for reasons including unexpected health, family or work problems. More information about how to apply for special consideration can be found at http://www.sau.uts.edu.au/assessment/consideration.html.

8. Marking scheme

Task	**Mark**
Show main window on launch	6%
Add a student	19%
Login	16%
Enrol	21%
Withdraw	4%
Remove a student	5%
Layout	7%
Style	7%
MVC architecture and FXML	5%
Correct package structure	1%
JavaFX observable properties and lists	4%
Attendance mode stored as ft/pt	2%
Correct use of exceptions	3%

Penalty for spoofing

If you spoof your answer to any verification point, the 1 mark awarded by PLATE for that verification point will be deducted, and a further penalty of 1 mark will apply.

There are two ways to spoof PLATE:

1. If you answer [y] for a verification point that your application does not actually support, that is a spoof. E.g. if you answer [y] that the UTS Logo is shown, but your application does not actually show the UTS Logo, that is a spoof.
2. If you answer [y] for a verification point that your application "seems" to support but only by hard coding the answer rather than using data from the model, that is a spoof. E.g. If you show the heading "Logged in as Bianca Sladen" by hard-coding

that exact string rather than fetching the student name “Bianca Sladen” from the model and incorporating that into the string, that is a spoof.

If you are unsure of whether you have spoofed a particular item, it is your responsibility to first check the FAQ to see if your question has already been answered, or ask the subject coordinator by email (Ryan.Heise@uts.edu.au) before the due date.

Penalty for lack of progress evidence

You must submit your progress to PLATE in increments of no larger than 25 marks. The penalty for making a submission to PLATE that is N marks higher than your previous best mark, where N > 25 is N marks. Here is an example:

Submission #1: 5 marks (OK. This is 5 marks higher than the previous best of 0)
Submission #2: 18 marks (OK. This is 13 higher than the previous best of 5)
Submission #3: 15 marks (OK)
Submission #4: 42 marks (“Just” OK. This is 24 higher than the previous best of 18)
Submission #5: 85 marks (**Penalty of 43**. This jump is 43 marks higher than 42)

To avoid a penalty, submit your progress in smaller increments. That is, you should not introduce more than 25 new [y] answers in your progress.txt file in a single submission.

When you submit to PLATE, always check that your marks were counted. If you entered some [y] answers but you did not include the Java source files in your submission, then your marks will not be counted and you will receive zero.